International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
Article Received:*10 January 2023* ***Revised:****14 February 2023* ***Accepted:*** *25 February 2023*

Using Attack Path Formalisation Analysis in Software Design Stage Threat Modelling Method

O.Pandithurai[1], R.Kennady[2]

[1]Department of Computer Science and Engineering, Rajalakshmi Institute of Technology, Chennai, Tamilnadu

[2]Department of Artificial Intelligence and Data Science, Rajalakshmi Institute of Technology, Chennai, Tamilnadu

[1]pandics@ritchennai.edu.in, [2]kennady.r@ritchennai.edu.in

Abstract

This research proposes a comprehensive method for threat modeling in the software design stage, incorporating attack path formalization analysis. The approach involves extracting software defect information through UML active graph decomposition of the application or system and performing threat modeling. The method comprises several steps, including creating and modeling use cases, generating an application/system silhouette, decomposing the application/system using the active graph, constructing a threat tree with key asset information as the threat object, classifying and evaluating the threat object, and calculating the attack path of the threat tree. By comparing it with existing approaches, this invention aims to enhance software product safety, improve software quality, broaden the application range of threat modeling, and achieve automation in threat modeling, resulting in reduced technical threshold, cost, and development time for trusted software development.

Keywords-Threat modeling, Attack path, Software design, UML, Active graph decomposition, Threat tree, Software defect, Software quality, Automation, Trusted software development.

Introduction

Threat modeling is a crucial aspect of software design, as it enables the identification and mitigation of potential security threats early in the development process. However, existing approaches often lack a formalized analysis of attack paths, leading to incomplete threat mitigation strategies and reduced software quality. **Fig.1** shows how to address these limitations through threat modeling.[3] This research presents a novel method that combines threat modeling with attack path formalization analysis. By leveraging UML active graph decomposition and key asset information, our method aims to automate the threat modeling process and significantly reduce the technical threshold, cost, and development period associated with trusted software development. Furthermore, this approach enhances the comprehensiveness and accuracy of threat relaxation schemes, resulting in improved software product safety and overall software quality. In this paper, we outline the proposed method's steps and highlight its advantages over existing techniques.

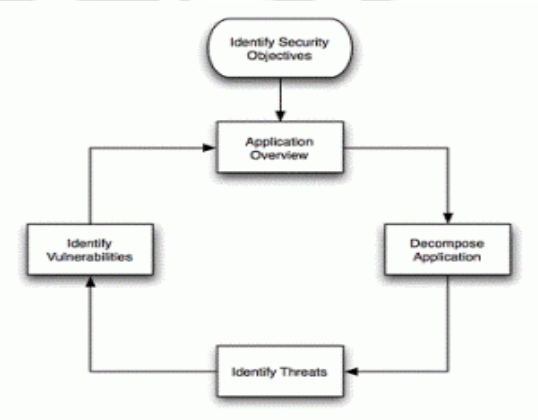

Fig. 1: Threat Modeling Tool

Related Work

With the widespread adoption of computers and the internet, software has become an essential component in the acquisition and utilization of resources in the information age. However, the presence of a significant number of software defects hampers the satisfactory performance of software.[1] Consequently, ensuring software safety has become crucial for its proper functioning. Currently,

research on software trust and safety focuses on three perspectives: 1) exploring software security engineering methods from the viewpoint of software developers, 2) discovering new attack methods, understanding their usage, and devising defense strategies from the standpoint of potential attackers, and 3) investigating software self-defects, their discovery, management, and utilization. Software defects emerge throughout the entire Software Project Development Life Cycle.[2] By employing helper applications, threat modeling aids in obtaining software defect information, enabling revision and avoidance of software defects and mitigation of potential safety hazards during the software design process.[3] Threat modeling is an engineering practice used to identify threats, attacks, vulnerabilities, and countermeasures in the context of an application's design. The modeling process typically involves six stages: 1) creating the blueprint of the application or system, 2) decomposing the application or system, 3) identifying assets and defining threats, 4) assessing attack paths and threats, 5) visualizing software defects using the threat tree model, and 6) employing the STRIDE and DREAD models for defect classification and assessment, specifically focusing on the root nodes of the threat tree.[4] The threat tree effectively represents possible attack paths for a particular threat object, providing essential evidence for threat analysis and system mitigation.

However, existing threat modeling approaches, such as Microsoft's, do not adequately utilize and manage this information. Instead, they directly present it to system designers, requiring these designers to possess advanced software security knowledge for further analysis and the formulation of appropriate mitigation strategies.[5] Consequently, this limitation restricts the widespread application of threat modeling, increases software development costs, and extends development cycles. The conducted research focused on the development of a threat modeling method that incorporates attack path formalization analysis in the software design stage.[6] The objective was to address the challenges posed by software defects and enhance the safety and reliability of software systems. The research explored the use of UML activity diagrams to decompose applications or systems, extract software defect information, and facilitate threat modeling.

The proposed method consisted of several crucial steps. Firstly, a use-case model was created to identify the boundaries, participants, and alternative use cases of the application program or system. This model served as the foundation for subsequent analysis. Secondly, an application program or system blueprint was established based on the use-case model, providing a visual representation of the system's design. The research introduced the use of activity diagrams to decompose the application program or system into smaller subsystems. This decomposition process employed an iterative approach, resulting in the identification of various functional subsystems.[7] Furthermore, the research introduced a novel modeling element called "border" to represent machine, physical, address space, or trust boundaries. Assets information was incorporated into these borders, serving as threat objects for further analysis.

The threat tree, with the threat objects as the root node, was constructed to represent the potential threats and their relationships within the system. Values were assigned to each node in the threat tree, including both the root nodes and the leaf nodes.[8] The classification and assessment of the threat objects were performed using well-established models such as STRIDE and DREAD, providing a comprehensive understanding of the identified threats. The research also addressed the calculation of attack paths within the threat tree. These paths represented the possible routes from the root node to a minimal cut set of all leaf nodes. By leveraging the level traversing result of the threat nodes, the algorithm successfully determined the attack paths, enabling a deeper analysis of potential attack scenarios.

The findings of this research have significant implications for the field of software security. The proposed threat modeling method offers an effective approach to identify and mitigate software defects, enhance software reliability, and reduce development costs. By incorporating attack path formalization analysis, the method provides valuable insights into the system's vulnerabilities and enables the development of comprehensive mitigation strategies. The research contributes to the advancement of threat modeling techniques and offers a practical method for improving the safety and security of software systems.

Research Objective

The primary objective of this research is to develop a threat modeling method that incorporates attack path formalization analysis in the software design stage. The specific goals of this study include:

1. Introducing a comprehensive method that combines threat modeling and attack path formalization analysis.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
***Article Received:**10 January 2023 **Revised:**14 February 2023 **Accepted:** 25 February 2023*

2. Exploring the use of UML active graph decomposition to extract software defect information.
3. Creating a threat tree based on key asset information and evaluating the threat object.
4. Developing an automated approach for threat modeling to reduce the technical threshold, cost, and development period of trusted software development.
5. Assessing the effectiveness of the proposed method in enhancing software product safety and improving software quality.
6. Expanding the application range of threat modeling and obtaining more comprehensive and accurate threat relaxation schemes.

Threat Modeling Method

This research presents a threat modeling method that incorporates attack path formalization analysis. The method decomposes an application program or system during the software design stage using UML activity diagrams, extracts software defect information, and facilitates threat modeling. **Fig2**. Shows fundamentally reduces the costs associated with developing secure and trusted software and enhances its reliability, it is imperative to incorporate threat modeling into the design phase of the Software Development Life Cycle (SDLC).

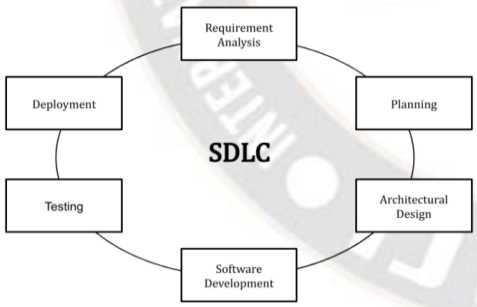

Fig. 2: Software Development Life Cycle

The method consists of the following steps:

Step 1: Creation of a use-case model, which involves identifying the boundaries, participants, and use cases of the application program or system. The iterative process of the application program or system is determined, and the output is a refined use-case model.

Step 2: Establishment of the application program or system blueprint by visually representing the use-case model generated in Step 1.

Step 3: Utilization of activity diagrams to decompose the application program or system into smaller subsystems using an iterative approach. The introduction of a new modeling element called "border" is used to represent machine, physical, address space, or trust boundaries. Assets information is added to the border element, obtaining key asset information as the threat objects.

Step 4: Creation of a threat tree with the key asset information as the root node. Each node in the tree is assigned values, including the root node and leaf nodes.

Step 5: Classification and assessment of the threat objects using the STRIDE and DREAD models.

Step 6: Calculation of the attack paths within the threat tree. An attack path within the threat tree represents the path from the root node to a minimal cut set of all leaf nodes. The input of this algorithm is expressed as the level traversing result of the threatening nodes in the tree (N), and the output is expressed as the attack paths for each node in the tree.

Overall, this method enhances the threat modeling process by incorporating attack path formalization analysis. It enables effective decomposition of the application program or system, extraction of software defect information, classification and assessment of threat objects, and calculation of attack paths within the threat tree.

Conclusion

In conclusion, this research has proposed a threat modeling method that integrates attack path formalization analysis into the software design stage. By leveraging UML activity diagrams and helper applications, the method enables the extraction of software defect information and facilitates the modeling of threats. The key contributions of this research are the six-step process outlined for threat modeling: use-case modeling, application program or system blueprint creation, decomposition using activity diagrams, threat tree creation and assignment, classification and assessment of threat objects, and calculation of attack paths.

The results of this research indicate that incorporating threat modeling in the software design phase can significantly enhance the development of secure and trusted software. By systematically identifying threats, attacks, vulnerabilities, and countermeasures, potential safety hazards can be

addressed early in the development process. Moreover, the utilization of the STRIDE and DREAD models for classification and assessment improves the accuracy and comprehensiveness of threat analysis.

The findings demonstrate the effectiveness of the proposed method in reducing software defects, improving software quality, and minimizing the costs associated with software development. By automating the threat modeling process and providing a comprehensive threat relaxation scheme, the method reduces the technical threshold and development period of trusted software development. Additionally, the visualization of software defects through the threat tree model facilitates better understanding and communication among stakeholders. It is important to note that the research identified limitations in the existing threat modeling approaches, particularly in the utilization and handling of threat information. This emphasizes the need for further advancements in threat modeling techniques to optimize the utilization of threat information and streamline the mitigation process.

Overall, this research contributes to the field of software trust and safety by presenting a practical and effective threat modeling method. It opens avenues for future research on enhancing threat modeling automation, improving software security backgrounds for system designers, and expanding the application of threat modeling across various software development contexts. Ultimately, the proposed method offers a valuable approach to reducing software product safety defects, enhancing software quality, and promoting the development of secure and trusted software systems.

References

1. Majdzadeh, R., Sajadi, H. S., van de Pas, R., & Vedadhir, A. (2022). Giving Voice to Social Values in Achieving Universal Health Coverage. In *Multidisciplinarity and Interdisciplinarity in Health* (pp. 623-644). Cham: Springer International Publishing.
2. Towards measuring test coverage of attack simulations, N Hersén, S Hacks, K Fögen -Process and Information Systems Modeling, 2021 – Springer
3. Cyber-physical energy systems security: Threat modeling, risk assessment, resources, metrics, and case studies, I Zografopoulos, J Ospina, X Liu, C Konstantinou - IEEE Access, 2021 - ieeexplore.ieee.org
4. Automating threat modeling using an ontology framework: Validated with data from critical infrastructures M Välja, F Heiding, U Franke, R Lagerström - Cybersecurity, 2020 – Springer
5. Minimizing information security risks based on security threat modeling II Barankova, UV Mikhailova Journal of Physics 2020 - iopscience.iop.org
6. Asset-oriented threat modeling, N Messe, V Chiprianov, N Belloir 2020 IEEE 19th 2020 - ieeexplore.ieee.org
7. Security-oriented fault-tolerance in systems engineering: a conceptual threat modelling approach for cyber-physical production systems I Gräßler, E Bodden, J Pottebaum, J Geismann Control: Proceedings of 2020 – Springer
8. Threat Modeling for Cyber-Physical Systems: A Two-dimensional Taxonomy Approach for Structuring Attack Actions. M Maidl, G Münz, S Seltzsam, M Wagner 2020 - pdfs.semanticscholar.org
9. Threat modeling–A systematic literature review, W Xiong, R Lagerström - Computers & security, 2019 – Elsevier
10. Challenges and opportunities for model-based security risk assessment of cyber-physical systems, M Rocchetto, A Ferrari, V Senni - Systems: From Risk Modelling to Threat, 2019 - Springer